TR-839
DAAG-53-76C-0138

December 1979

A GENERAL-PURPOSE SOFTWARE PACKAGE FOR ARRAY RELAXATION

Russell C. Smith
Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND
20742

TR-839 December 1979
DAAG-53-76C-0138

A GENERAL-PURPOSE SOFTWARE PACKAGE FOR ARRAY RELAXATION

Russell C. Smith
Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

Abstract

Probabilistic relaxation as an image processing tool is becoming increasingly common. This report examines two versions of probabilistic relaxation, that initially proposed by Hummel, Zucker, and Rosenfeld, and the version recently introduced by Peleg. A software package is presented which allows either method to be applied to probabilistic images quickly and easily. The package makes full use of the power available under Bell Laboratories' UNIX operating system running on a PDP11/45 computer. Modular in form, it frees the researcher from the tedium of hand coding relaxation processes for each variation of relaxation tested. The application of relaxation to a threshold-like gray level modification scheme demonstrates the utility of the package.

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order No. 3206) is gratefully acknowledged. The author also acknowledges comments and suggestions received from many people, including Alan Danker, Sanjay Ranade, Gyuri Fekete, and Azriel Rosenfeld.

Table of Contents

Table of Contents (continued)

List of Tables

List of Figures

1. Introduction

Iterative techniques in image processing resemble the processes that may take place in natural vision systems. Noise cleaning, object extraction, line and edge extension, all lend themselves to repetitive operations. A general purpose iterative technique, probabilistic relaxation, has been found to be useful in a wide range of applications. It can be used for tasks ranging from very low level processing such as noise cleaning, to very high level processes such as scene labeling. This thesis examines the formulation for probabilistic relaxation, introduces a general purpose software system which allows easy and quick experimentation using probabilistic relaxation techniques, and presents applications of gray level relaxation to the extraction of objects from images.

2. Probabilistic Relaxation

Quite often in digital image processing operations in the immediate neighborhood of each pixel are performed in order to determine certain properties which the pixel may possess (gray level, edge strength, etc.). These local properties are then used to classify the pixels and hence aid in the extraction of information from the image. Unfortunately, local operations are highly sensitive to the presence of noise, especially so if the classification for each pixel is independent of those of its neighbors. One method used to correct for the effects of noise on local operations is to iteratively update pixel classifications based on the classifications of neighboring pixels, allowing one iteration's results to be reinforced or attenuated at the succeeding iteration. When applied over an entire image in a parallel fashion this method is called "array relaxation".

2.1. Hummel and Zucker's Relaxation Method

In [1] a method is proposed whereby each object (e.g. each point) in a scene has a set of possible labels, the weight of each label lying between 0 and 1, and the sum of the weights of all possible labels for an object being 1. Hence, the weight of a label L of an object can be thought of as the probability that L is the correct label for that object.

Given this probabilistic labeling of the pixels in an image we now want to iteratively update the probabilities. Certain properties of the relaxation updating rule become desirable. The probability of a label L of a pixel should be increased if those labels of the pixel's neighbors which are highly compatible with label L have high probabilities. The probability of label L should be decreased if the high-probability labels of the neighbors are incompatible with L. If the neighbors' labels have low probabilities, their effect on label L should be minimal, regardless of the compatibility between the labels. Figure 1 presents these properties in tabular form.

The compatibilities between the neighboring pixels' labels can take on values in the range [-1, 1], where a negative value indicates incompatibility, a positive value indicates compatibility, and a value near or at zero indicates that the labels should have little or no effect on one another. (Other ranges for the compatibilities are equally valid; see Section 2.3). The formula given below behaves the way we would want the change in a particular label's probability to behave.

$$q_i^{(k)}(L) = \sum_j d_{ij} \sum_{L'} p_j^{(k)}(L') r_{ij}(L,L') \qquad (1)$$

Here the d factor is a weighting of the point's neighbors' contributions (the sum of the weights is 1) and the r(L,L') factor is the compatibility of label L at pixel i

with label L' at pixel J.

From this can be defined the relaxation updating rule itself which is applied in parallel to every pixel in an image using the results of the k-th iteration to compute those for iteration k+1:

$$p_i^{(k+1)}(L) = \frac{p_i^{(k)}(L)[1 + q_i^{(k)}(L)]}{\sum_{L'} p_i^{(k)}(L')[1 + q_i^{(k)}(L')]} \qquad (2)$$

Here the factor [1+q(L)] keeps the probability of any of the pixel's labels nonnegative and the denominator is used to normalize the label probabilities.

It can be seen that this formula does indeed exhibit the desirable properties presented in the preceding paragraphs and in Figure 1. Of course, other updating rules can be defined which possess these properties (e.g. see Section 2.3 or [2]).

Though the formula is simple, it has proven to be quite powerful in a wide range of applications. In [3], for example, interior, edge, and noise points for dot clusters have their strengths reinforced according to the probabilities of neighboring interior, noise, and edge labels. In this example the compatibility coefficients were computed using distance as a key parameter (i.e. a noise point far away from an interior point would tend to be either more compatible than one close by, or, outside a certain range, it would become irrelevant).

An interesting example of noise cleaning is given in [4]. Here the initial probabilities represent a normalized function of the observed gray levels. The compatibility coefficients used are based on the differences of possible labelings. When the resulting formula is simplified a weighted average of the neighborhood's label probabilities results, easing the computation load.

An example involving pixel classification based upon multispectral data is given in [5]. The initial probabilities are computed by clustering the points and using a function of the distance of a point from each cluster mean as the initial label weight. The compatibility coefficients were computed using mutual information, an automatic process discussed in Section 2.2. Here the improvements gained from the relaxation method were found to be considerably better than those obtained from iterated pre- and post-processing methods.

References to and summaries of other examples from the wide range of applications of relaxation in image processing are given in [6].

2.2. Compatibility Coefficients

In the preceding section a general purpose relaxation formula was presented. Based upon initial probabilities and compatibilities between labels good results can be obtained.

However, the highly important compatibility coefficients (essentially the heuristic in the relaxation process) were found to be computed in many different ways, depending on the type of data being used. What is needed is a general method to compute these coefficients. In [7] exactly this has been done.

2.2.1. Correlations

Two methods of automatically computing compatibility coefficients are presented in [7]. One method, using statistical correlations between labels, was found to produce poor coefficients because labels which dominate in an image (as "background" would in an "object-background" labeling) would tend to be highly compatible with all labels and so mask out any beneficial effects of the relaxation process. This is corrected by weighting the coefficients by the probability that the labels do not occur, hence weakening coefficients involving dominant labels while having little effect on those coefficients involving rarely occurring labels. One inconvenience of this correction scheme is that points obtaining little or no information from their neighbors from one iteration to the next tend to have their "rare" label probabilities increased, simply because the point is considered to be its own neighbor and so reinforces itself. This, too, can be "corrected" by disregarding the self-support case.

The initial correlation coefficients can be expressed by:

$$r_{ij}(L,L') = \frac{[p_i(L) - \bar{p}(L)][p_j(L') - \bar{p}(L')]}{\sigma(L)\sigma(L')} \qquad (1)$$

where $p_i(L)$ is the initial probability estimate for label L at point i. $\bar{p}(L)$ is the average p(L) over all i, and $\sigma(L)$ is the standard deviation of p(L).

The weighting correction can then be applied giving the compatibility coefficients:

$$r_{ij}^{*}(L,L') = [1 - \bar{p}(L)][1 - \bar{p}(L')]r_{ij}(L,L') \qquad (2)$$

When a relaxation process using these coefficients was used to aid in curve enhancement the results obtained after many iterations did not appear to differ much from the results which a maximum likelihood classifier performed on the initial probabilities would obtain. That is, choosing the maximum of the initial label probabilities for each point would have done essentially as well. Though it is not clear whether this would hold in a variety of cases, correlations as coefficients suffer from the fact that "ad hoc" as opposed to analytical methods are used for their computation.

2.2.2. Mutual Information

The second method presented in [7] computes the compatibility coefficients based on the mutual information of the labels of neighboring points.

Initially, the probability of any point having label L is estimated by taking the average over the entire image of each point's label L probability:

$$p(L) = \frac{1}{N} \sum_i p_i(L) \tag{1}$$

where N is the number of points in the image. Similarly the joint probability of a point i having label L and its neighbor j having label L' is estimated by

$$p_{ij}(L,L') = \frac{1}{N} \sum_i p_i(L) p_{ji}(L') \tag{2}$$

where p_{ji} is a particular neighbor of point p_i. The conditional probability that point i is labeled L given that neighboring point j is labeled L' can then be estimated by

$$p_{ij}(L|L') = \frac{p_{ij}(L,L')}{p(L')} = \frac{\sum_i p_i(L) p_{ji}(L')}{\sum_i p_i(L')} \tag{3}$$

Now, the amount of information obtained as a result of being told that an event A occurred (with probability p(A) of occurring) is defined as

$$I(A) = -\text{Log } p(A) \tag{4}$$

Hence, the conditional information obtained if we know that B has occurred and we are told that A has occurred is

$$I(A|B) = -\text{Log } p(A|B) \tag{5}$$

The contribution of B to the information about A can then be expressed by the "mutual information"

$$I(A;B) = I(A) - I(A|B) \tag{6}$$

$$= \text{Log } \frac{P(A|B)}{P(A)} \tag{7}$$

It can be seen that this allows the correlations between events to be reflected in the values for $I(A;B)$: if A is positively correlated with B, $I(A;B)$ will be high; if A is negatively correlated with B, $I(A;B)$ will be small.

Using equations (1), (3), and (7) the mutual information coefficients can then be derived as

$$r_{ij}(L,L') = \text{Log } N \frac{\sum_i p_i(L) p_{ji}(L')}{\sum_i p_i(L) \sum_i p_i(L')} \tag{8}$$

These values can vary outside the range $[-1,1]$, but the instances outside the range $[-5,5]$ are so rare that they can be considered virtually impossible (indeed, when computing these values over a single small image, they are

impossible). So, the coefficients produced by (8) can be divided by 5 to obtain the correct range.

The use of mutual information values as compatibility coefficients produces good results in the curve enhancement example, results which are as good as those obtained using modified correlations. Section 4 reconfirms this with examples of thresholding using relaxation in which mutual information was used to compute the compatibility coefficients. The straightforward computations involved in mutual information and its analytical justifiability were major factors in its selection as the method of automatic coefficient computation available in the software package (see Section 3).

2.3. Peleg's Relaxation Method

A new formula for probabilistic relaxation is presented in [8]. This formula not only has the advantage of being analytically derived using probability theory, but also can be easily expanded to N-tuple interactions rather than interactions between an object (point) and a single neighbor (N-tuples are useful in handwriting analysis, for example, where each object's labels consist of the letters of the alphabet. N-tuple relaxation will not be discussed further in this paper.).

Peleg's relaxation method differs from that of Hummel and Zucker in two ways: (1) the initial probabilities which

are updated are directional, i.e., an object has its probabilities updated with respect to a single neighbor only, not all neighbors together; (2) a point is not considered to be its own neighbor, hence there is no self-support. A "post-processing" step over all the neighbor-relative probabilities (averaging) is used to derive the next iteration's nondirectional probabilities.

As with Hummel-Zucker relaxation we first have a value which acts as we would want a change in a label's probabilities to behave. We then multiply the point's label probability by this value to obtain the intermediate result:

$$q_{ij}^{(k)}(L) = p_i^{(k)}(L) \sum_{L'} p_j^{(k)}(L') r_{ij}(L,L') \qquad (1)$$

Note that this q value intrinsically takes into account neighbor j. The r(L,L') factor, still called a compatibility coefficient, is in fact actually derived along with the rest of the updating rule from probability theory. Quite similar to the mutual information coefficient, it is computed as:

$$r_{ij}(L,L') = \frac{p(L,L')}{p(L)p(L')} \qquad (2)$$

Like mutual information, these coefficients can be computed from the initial label probabilities and will remain static throughout the relaxation process.

To reconvert the q values to be between zero and one a standard normalization is done:

$$p_{ij}^{(k)}(L) = \frac{q_{ij}^{(k)}(L)}{\sum_{L'} q_{ij}(L')} \tag{3}$$

These updated label probabilities are still directional in nature. To derive nondirectional probabilities an average over all neighbors can be taken:

$$p_i^{(k+1)}(L) = \frac{1}{N} \sum_j p_{ij}^{(k)}(L) \tag{4}$$

Averaging, though perhaps not the optimal method of computing the probability estimates, was found in practice to produce better results than a computationally more complex normalized minimum function. Peleg's relaxation scheme as implemented in the software package uses averaging of pairwise estimates.

2.4. Discussion

Of the two relaxation methods presented in the preceding sections, that proposed by Peleg seems to be more strict in its derivation. Claims were made in [8] that it also seems to work slightly better than standard relaxation using two radically different domains. As can be seen in the application to thresholding in Section 4, relaxation according to Peleg does not necessarily perform better than standard relaxation in all cases. Both methods were

therefore implemented in the software package.

No optimal way to determine compatibility coefficients has yet been devised. Mutual information lends itself to easy application with probabilistic images but pays no regard to the actual meaning of the labels. It does, unlike general purpose hand computed coefficients, allow some image content information to be used in the relaxation process.

3. A General Purpose Array Relaxation Implementation

Image processing techniques based on probabilistic relaxation are becoming increasingly common. Each application has usually been uniquely implemented by the individuals doing the research, tailored to very specific needs. Due to the varying parameters in the problem at hand such as number of possible labels, size of a point's neighborhood and number of interacting relaxation processes, variations on the relaxation formula are often directly encoded in software along with initial probability estimate computation. As might be expected, this leads to a great duplication of effort. The software package presented herein allows each user to quickly create problem specific compatibility coefficients and relaxation programs, freeing him or her to concentrate on computing the initial probabilities and doing the actual processing of data.

3.1. An Overview

In order to make a software package general enough to be used for a wide variety of problems a determination must be made as to what should be allowed to change and what should remain static over any possible variation of problem definition. In the relaxation domain a number of items which should be allowed to vary are present. These include the image size, the number of labels, the number of neighbors a point possesses (i.e. the size of a pixel's neighborhood),

the relaxation method to use, and, in some cases, the number of interacting relaxation processes. Some of these lend themselves very easily to run time computation. Others, when computed at run time, tend to slow down the execution of the routine due to the accommodations necessary in the code. For this reason, some variables are allowed to change only up to the time of compilation (such as the relaxation method) while others can be varied at run time.

The package consists of:

a) programs to compute compatibility coefficients according to one of the two formulas presented in Section 2;

b) programs to compute one iteration of relaxation by the Hummel-Zucker or Peleg method;

c) a display program which will display that label of a point which has a probability greater than any other as a gray level;

d) an interactive neighborhood definition program which allows the user to set up a point's neighborhood to be any of the points within a maximum sized neighborhood;

e) an interactive program to allow the user to hand-compute the compatibility coefficients.

The automatic coefficient computation programs and the relaxation programs are compiled according to parameters which the user inputs.

3.2. Control Flow

The software package is implemented on a PDP11/45 computer running Bell Laboratory's UNIX timesharing operating system [9]. Utilizing top level Shell commands, Shell command files, and modular routines written in the C programming language, the package allows the user to create problem-specific programs in a relatively short time.

Like other operating systems, UNIX has a top level command interpreter. Under UNIX the interpreter, called the Shell [10], differs from most command interpreters in that it has the capability of modifying the environment in which commands run, even to the extent that commands themselves are not defined until run time. This is possible due in part to a number of programming mechanisms which are quite similar to those found in structured programming languages, such as variable assignment, conditional execution of commands through the use of the "if" statement, and the passing of parameters. The latter feature provides a gentle push to the programmer to modularize any software written for the Shell, i.e. to create top level commands each of which performs only part of the processing desired, making each part very much easier to debug and so speeding up the programming task. When combined with other commands in a Shell "command file" one obtains what is essentially a highly structured program, a Shell program. The relaxation software package is designed around this philosophy.

Modular in form, the package allows a user to create problem-specific relaxation programs very easily. Figure 2 illustrates the general flow of control. Initially the gross outline of the desired relaxation process is determined by arguments on the call to the top level Shell program "setup". The arguments to this program define the relaxation method to use (Hummel-Zucker or Peleg), the number of possible labels which a point may have, and optionally, the maximum number of columns and rows to be considered as containing a point's neighborhood (the default maximum neighborhood size being 3 by 3). The arguments also help determine the flow of control of the setup program, that is, whether a Shell subprogram for relaxation program creation or a small C program for package description will be run. The latter routine allows a first time user to sit down at a console and use the package with a minimal foreknowledge of the ways the package can be used.

If the user inputs (either keyed in or from a file) the correct syntax for the desired relaxation method then a Shell subprogram will be run. This subprogram, whether for the Hummel-Zucker or Peleg relaxation method, will initially run a C routine to construct a file of parameters for later use by the compilation phase. The parameters consist of the number of labels, maximum number of columns and maximum number of rows in a point's neighborhood, plus a set of "event" flags computed from the preceding three parameters to be used to cause a change in program execution when the

user is actually running the coefficient computation or relaxation programs. For example, when an image processing program is first started some initialization sequence, such as reading in a given number of rows of data, must usually be performed. Similarly, when operating in a neighborhood around a point care must be taken to remain within the image's boundaries. The calculated parameters ensure that there are no violations of the probabilistic image's boundaries.

Upon completion of the construction of the parameter file the Shell subprogram will enter a compilation phase to create the two main programs for the user.

One program created automatically for the user can be used to compute the compatibility coefficients according to formula (8) of Section 2.2.2 or formula (2) of Section 2.3 depending on the relaxation method chosen. The coefficient computation program, though static with regard to the number of possible labels, allows both the image size and the number of neighbors each pixel has to vary. These two seemingly minor attributes none the less contribute greatly to the utility of both this program and the package in general. A user may run the coefficient computation program on a large image, using the resulting coefficients in the relaxation program on smaller images. Additionally, comparisons can be made of the differences between coefficients produced from large and small (though of

similar content) images. Compatibilities computed over different neighborhoods can also be compared to those analytically derived (in the two label application presented in Section 4, for example, a neighbor to the left of a pixel should have the same coefficients as one to the right).

The program produced by this part of the compilation phase is placed in the user's current directory under the name "*compat", where '*' is either 'h' or 'p' depending on whether Hummel-Zucker or Peleg relaxation is chosen.

The Shell subprogram will next construct and compile the relaxation routine. This routine will be an implementation of formulas (1) and (2) of Section 2.1 or formulas (1), (3), and (4) of Section 2.3, again depending on the relaxation method chosen. Items allowed to vary at run time are image size, neighborhood definition, and, if the Peleg method is chosen, the number of interacting relaxation processes (More than one process is desired in cases such as an edge-interior combination, where the presence of a neighboring point with a strong edge label should have influence over the interior-exterior labeling of the current point.). Multiprocess relaxation is identical to that with a single process until the final normalization is performed (formula (4), Section 2.3). At this point each label is normalized only with respect to the process set to which it belongs.

The relaxation program produced is placed in the user's current directory under the name "*relax" ('*' = 'h' or 'p'). Each time this program is run the input probabilistic image will be replaced by one produced by one iteration of the selected relaxation method.

Upon completion of the compilation phase the Shell subprogram will return to the main program "setup" which will display a message indicating successful completion and return control to the top level UNIX system.

The user has two options available with regard to running the programs produced by "setup". They may be directly invoked at the top command level (thus computing one iteration of relaxation) or they may be installed in a Shell program which may contain a programmed loop to compute many iterations. In either case the programs may be run in the foreground (the user must wait for a program to finish before doing any other processing) or in the background (the user is immediately free to do some other task).

3.3. Definition Enhancement and Display Programs

The preceding section described the automatic creation of coefficients and relaxation programs. These programs are sufficient for experiments in relaxation on images. However, the software package contains programs which can enhance those presented previously, as well as allow a wider domain of problems to be more easily

investigated.

Occasionally a user wants the neighborhood of each pixel to be defined as something other than an m by n rectangle. A program provided in the package allows the user to interactively define a point's neighborhood to consist of any of the points within a maximum neighborhood size (not necessarily all the points). A variable neighborhood definition permits the user to quickly and easily test the effect of different neighborhoods on the results of relaxation. For example, the difference between four and eight neighbor reinforcement can be readily checked. Additionally, as in [11], an unusually shaped neighborhood can be used to detect and enhance particular types of regions.

If mutual information is not the desired method for compatibility coefficient estimation a program in the package allows the coefficients to be interactively defined for each neighbor of a pixel. Since the coefficients are the heuristic behind the relaxation process, allowing them to be "hand-tuned" greatly increases the information attainable by the user as to the effects of relaxation. Section 4.2.4 examines the use and effect of hand-computed compatibility coefficients.

After one or more iterations of relaxation have been applied to an image a user usually would like to determine the effect on the initial image. One measure of the effect

of relaxation is to note the change in entropy of an image from one iteration to the next. Graphic illustrations of the entropy of probabilistic images following relaxation are given in [12]. Another way to determine the effect of relaxation is to actually display the probabilistic image as a gray level (or color) picture. This can be done by taking the maximum valued label of each point and displaying it as a gray level. The package contains a program which will convert a probabilistic image into a gray level picture. If there are two labels per pixel the maximum label for each pixel will be displayed as a varying gray level depending on label strength (one label, if maximum, will be converted to the range gray through black, the other, if maximum, to the range gray through white). The figures in Section 4 illustrate this conversion. If there are more than two labels per pixel the maximum label will be displayed as a constant gray level regardless of label strength (each label has a distinct gray level initially assigned to it which will be displayed if the label is the maximum over all the point's labels). [5] has figures illustrating multilabel conversion.

3.4. Discussion

The software package has been used by a number of researchers studying aspects of the use of probabilistic relaxation on images. Variations in the problems studied would have previously required considerably more time to

create task-oriented relaxation programs. Section 4 examines one of these applications: the extraction of objects from backgrounds using gray level based relaxation.

4. Thresholding Using Relaxation

Thresholding can be used on images to extract objects from their backgrounds. If an image is noisy, however, the results obtained by thresholding will also tend to be noisy. In addition, if there are regions in the image (unbounded by edges) having fluctuations in gray level which cross the threshold, they may also be extracted along with the objects. Relaxation can be used to improve on the results obtained by thresholding.

4.1. Light/Dark Relaxation

In the simplest method of thresholding by relaxation each pixel is initially assigned a "light" and "dark" probability based on its gray level (so Light/Dark relaxation in this case involves two-label classification). These probabilities are than iteratively updated based on the probabilities at the eight immediately neighboring points. In order to threshold we would want light to reinforce light and dark dark. Hence noise points, which are not similar to their neighbors, tend to have their label probabilities adjusted in such a way as to become more consistent with those of their neighbors, while all points are shifted to one of the two extremes of light and dark. Eventually, points of a light object have their light probabilities become uniformly high (and vice versa), allowing thresholding to yield considerably better results.

Let LOW be the lowest gray level in an image to be thresholded by relaxation. Similarly let HIGH be the highest gray level and assume that a point P has gray level GL so that

LOW .le. GL .le. HIGH for all P.

Taking LOW as corresponding to the dark end of the gray level range we can then estimate the probability that P is dark by

p(DARK) = (HIGH - GL) / (HIGH - LOW)

and the probability that P is light by

p(LIGHT) = (GL - LOW) / (HIGH - LOW)

= 1 - p(DARK)

Given this initial probability estimate of the labels of an image's points the computation of the compatibility coefficients can be automatically performed by the package according to either formula (8) of Section 2.2.2 or formula (2) of Section 2.3 depending on the relaxation method chosen.

It should be noted that the compatibilities between neighboring points' labels should be symmetrical. Indeed with an ideal image there would be only three distinct numbers for coefficients (LIGHT!LIGHT, DARK!DARK, and LIGHT!DARK). However, due to the fact that the coefficients

are computed from the nonideal images themselves, the values can vary depending on direction. The differences between directional values should still be slight for any image which contains either single objects with edges in all directions or many variously oriented but thin (and so essentially unidimensional) objects. An example of the former is ink splattered on a piece of paper; of the latter, thin lines drawn in random orientations.

Both relaxation methods were applied to a group of four images having varying content and gray level ranges: a signature, chromosomes, a LANDSAT picture of clouds over water, and a FLIR image of a tank (Figures 5 through 8. Because each histogram has been rescaled, its shape, not the individual gray level bin values, is significant). All the images have gray levels that are broadly distributed over the gray level range, a fact that is crucial to the simple initialization scheme described on the preceding page. Had there been a point or group of points sufficiently light or dark so as to cause most of the other points in the image to fall in the same half of the gray level range, errors in classification could have resulted. Figure 3 illustrates this problem. Note that even though there are two distinct peaks in the histogram, the presence of one noise point has forced all other points to be labeled "light", so that the relaxation process would degrade the initial probabilities. Section 4.2 discusses initialization procedures which can overcome this problem.

With both relaxation methods the compatibility coefficients were computed using the package-supplied mutual information program. They are shown in Tables 1 through 8. As expected there are slight variations in the values for the different directions of the neighbors, but it can be seen that these variations are relatively slight. In addition, note that the coefficients obtained from one image are quite similar to those obtained from any other image. In [7] results of curve enhancement experiments showed that coefficients produced from one image could be used with a relaxation operator on an entirely different image as long as it contained a "reasonable" set of curves. Two label gray level compatibility coefficients behave similarly.

Figures 9 through 16 show the results of eight iterations of relaxation for each image using both Hummel-Zucker and Peleg relaxation. Note that with both methods the thin lines in the signature tend to thicken and the tank's interior tends to fill in (similar effects on the cloud and chromosomes are not as readily discernable). This is a result of using mutual information coefficients; infrequent label pairs have higher mutual information. That this is an undesirable effect is open to question. Noise points are still eliminated, points with high probability LIGHT labels surrounded by similar points are reinforced (and vice versa), and points along the objects' edges tend to go either way. Only in those cases in which a point has one or more neighbors with the same label (but still in the

minority) will filling take place. An argument could be made that for a threshold-like scheme this may indeed be the effect desired.

On examination of the figures one finds that the Hummel-Zucker relaxation method appears to produce results more quickly than that of Peleg. This may be an artifact of the self-support present in the Hummel-Zucker method combined with the simple nature of the processing. However, if one disregards the number of iterations, both relaxation methods produce similar effects on the images.

With both relaxation methods the discrimination between light and dark regions becomes more distinct from one iteration to the next. As the histograms show, the points gradually shift toward one of the two gray scale extremes, creating two spikes with a nearly empty valley. Empirical tests have shown that the points represented by the valley are those which are on the edges of the objects where label reinforcement is expectedly not as strong, while the peaks represent those points surrounded by high probability similarly labeled points. Noise points, present as thin irregular streaks in the tank image, for example, have had their label probabilities shifted towards those of their neighbors, and, in general, are represented on the inside shoulders of the two peaks.

The results after the arbitrarily chosen eight iterations demonstrate that relaxation is a viable method of

producing a threshold-like effect. The simple initial probability estimation program combined with the software provided by the package were sufficient to test the effects of probabilistic labeling. However, it was shown that the initialization procedure was prone to errors. The next section examines variations of light/dark relaxation, some which are designed to overcome this problem.

4.2. Alternate Light/Dark Versions

Gray level relaxation can be used to aid in the extraction of objects from a background even when the image contains considerable noise. Based on an initial label probability estimate at each point the results after a few iterations are quite good. But the initialization process itself may not correctly take into account idiosyncrasies of the image at hand. This and the following section examine alternate methods of initial probability estimation.

4.2.1. The Histogram Mean

The initialization method presented in the previous section was extremely vulnerable to misclassification due to the use of the midpoint of the histogram as the initial light/dark transition. One initialization variation which can largely avoid similar misclassifications is to use the mean of the histogram as the light/dark transition point.

Let LOW, HIGH, P, and GL be as defined in the previous section and let M be the image's mean gray level. If GL is less than M then the probability that P is dark can be estimated by

$$p(DARK) = .5 + (.5(M - GL) / (M - LOW))$$

and the probability that P is light by

$$p(LIGHT) = 1 - p(DARK).$$

Similarly, if GL is greater than M then the probability that P is light is

$$p(LIGHT) = .5 + (.5(GL - M) / (HIGH - M))$$

and the probability that P is dark is

$$p(DARK) = 1 - p(LIGHT).$$

If GL is the same as M both formulas give equal values for p(LIGHT) and p(DARK).

This initialization scheme will correctly handle strong biasing of an image's histogram caused by bins which are insignificant but radically different in gray level.

4.2.2. Gaussian Fitting

An initialization method which will also correct for a skewed histogram was examined in [12]. Gaussian curves are fitted to the image's histogram. Each label probability is

then computed from the Gaussian curves. Unlike the previous scheme, this will allow label probabilities to more closely represent the region membership of a point, especially in those cases in which one region greatly exceeds the other in area. Figure 4 illustrates this. Had the histogram midpoint or mean been used as the light/dark transition, many points would have been falsely biased toward a LIGHT labeling. Using Gaussian curves produces more correct initial label probability estimates.

Gaussian curve fitting has the advantage that it can be trivially extended to more than two labels, for example, allowing images where there are objects both darker and lighter than the background to be handled. It has the disadvantage of needing at least a bimodal histogram in order to fit more than one curve, a problem not present in the previous two methods. Images similar to the tank might cause this initialization method to fail.

4.2.3. Multilabel Relaxation

A multilabel version of gray level relaxation which did not produce good results used one label for each gray level present in the image. Assuming that GL is point P's initial gray level, then P's label probabilities were estimated as

$$p(GL) = .5$$

$$p(GL+1) = p(GL-1) = .2$$

$$p(GL+2) = p(GL-2) = .049$$

$$p(ALL-OTHERS) = .002 / N$$

where N is the number of other possible labels, hence p(ALL-OTHERS) is some small number greater than zero. Note that this initialization will define labels outside the gray level range. This causes no problems because it is guaranteed that the probabilities of these labels will only be attenuated from one iteration to the next.

This method would theoretically allow points of similar gray level (not necessarily identical) to reinforce one another, eventually causing a shift not to the two gray level extremes, but to the levels which best represent the gray level of the region to which the point belongs (not unlike converting a histogram into many spikes). However, experiments showed that a very slight noise cleaning was the only effect that could be noticed after many iterations. This can probably be attributed to the lack of any high initial probabilities; it would take much longer for the change to become noticeable. Variations, including setting the probabilities of labels within +2 or -2 levels to the same initial estimate, did not provide much improvement.

Of the variations on light/dark relaxation, that using the midpoint of the image's histogram for initialization purposes is certainly the least complex, though perhaps prone to noise-caused misclassification. Using the mean will allow most cases to be handled with the

possibility that some points may be incorrectly biased toward the wrong label. It requires only slightly more computation than that using the midpoint. Fitting Gaussian curves more closely matches a gray level range with a region in the image, with a large increase in the computational load. Though it can be easily extended to multilabel relaxation, two curves do have to be fit to the histogram, a requirement that cannot always be met. The three methods all have their good and bad points. Choosing which is "correct" may be an image-relative decision.

4.2.4. Hand Computed Compatibility Coefficients

In all the preceding versions of gray level relaxation the compatibility coefficients were computed using the package-supplied mutual information program. The coefficients, though similar for the eight possible directions, were not identical as they would have been with an ideal image. In order to determine the effects of symmetry and variations in compatibility strength an experiment was run using manually computed coefficients. Figure 17 shows eight iterations of the Peleg relaxation method on the tank image using hand defined compatibility coefficients. In this example the LIGHT!LIGHT and DARK!DARK compatibilities were considered to be much higher than those of LIGHT!DARK. It can be seen that light and dark regions rapidly approach the two extremes while those regions with both light and dark points shift towards the more dominant

labeling. The resulting discrimination between light and dark regions is excellent but the physical form of the tank is no longer easily recognizable. This result is very similar to actually thresholding a slightly blurred version of the original image. When the LIGHT!LIGHT and DARK!DARK compatibilities were considered to be close to those of LIGHT!DARK the end results obtained were virtually identical, though more iterations were required.

Manually computed compatibility coefficients require that very general information about the problem at hand be applied to a wide domain of images with the possibility that the results may be poor in certain cases. The use of image-specific information allows variations in image content to be taken into account. Though mutual information coefficients may not be the optimal way to encode this information, they are general enough to work well in different domains.

4.3. Borderness

The gray level relaxation schemes presented in Sections 4.1 and 4.2 operated on probabilistic images whose initializations were based on histogram content, the actual configuration of the light and dark pixels in the original image being ignored. The compatibility coefficients, if they were computed from the image itself, could only partially contain image-specific information. The results of gray

level relaxation can be improved if the initialization is based on both histogram and image. This can be accomplished by letting the light label or dark label probabilities be reinforced on initialization if they are on the light or dark side (respectively) of an edge. This method of computing the initial light/dark label probabilities would tend to deemphasize those regions in an image which are unbounded by edges.

To produce this initialization a "borderness" image is constructed from the initial gray level image. First a set of masks

```
-1  0  1      0  1  1      1  1  1
-1  P  1     -1  P  1      0  P  0     . . .
-1  0  1,    -1 -1  0,    -1 -1 -1
```

is applied to each point. The edge value obtained for each mask is added into those neighbors of point P whose mask value is 1. This is done for all masks at each point. The resulting borderness image will have high values only on the light side of edges.

This borderness image can then be combined with the gray level image to yield a probabilistic image whose LIGHT label probabilities will be emphasized on the light sides of edges. Let the gray level probabilities be computed as in the previous section and let PL be the value of point P's LIGHT label. Let B be the borderness value at point P and BMAX be the maximum borderness value of the image. The new

light/dark label probabilities can then be computed as

$$p(\text{LIGHT}) = (A * PL) + (1 - A)(B / BMAX)$$

$$p(\text{DARK}) = 1 - p(\text{LIGHT})$$

where A is between zero and one.

Using this initial probabilistic image with the compatibility coefficients obtained from the gray level image produced results which were better than relaxation on the gray level image alone. Figure 18 shows the original and eight iterations of light/dark relaxation applied to an image of a blob. Regions in the background have been extracted along with the blob even though they are not bounded by strong edges. Figure 19 shows the original and eight iterations of the combined gray level and borderness images using values of .5, .25, and .01 for A. The blob is strongly reinforced while the background gray level fluctuations have largely been suppressed. Using image-specific information in the probability initialization has improved the results of the relaxation process.

4.4. Discussion

Objects can be extracted from backgrounds using relaxation processes. Section 4 presented different versions of single process light/dark relaxation which performed quite well. The use of a light/dark relaxation process combined with an edge/no-edge process has been found

to produce results better than those using single processes alone [13]. Extending this concept, multiprocess relaxation using more than two interacting processes could extract objects from an image even better. Investigations into many relaxation variations are currently in progress.

5. Concluding Remarks

This thesis has examined the concept of probabilistic relaxation and presented a software package which allows the researcher to quickly obtain problem-specific relaxation programs. A variety of applications involving gray level relaxation demonstrated the versatility of both relaxation processes and the software package. Relaxation processes simulate actions which natural vision systems are believed to perform (such as the eye filling in objects from their edges) and permit investigation into the use of multilayer arrays (pyramids). It is virtually certain that, with the advent of cheap processing hardware, relaxation-like software will become very common.

Figures and Tables

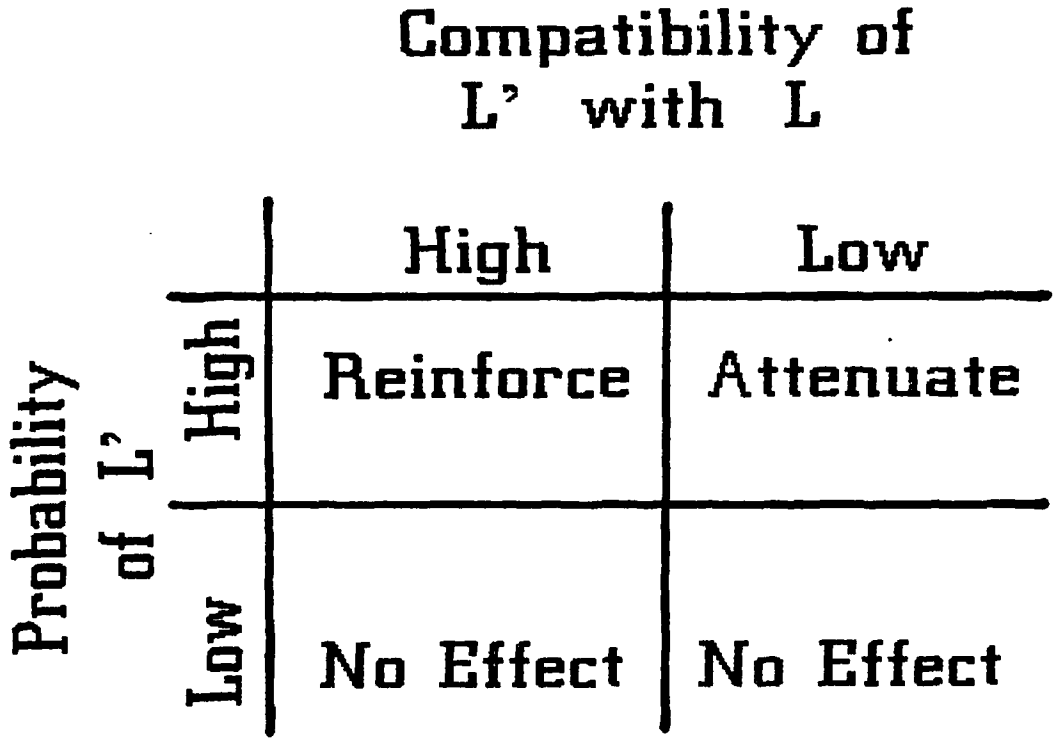


Fig. 1 Updating Rule Properties

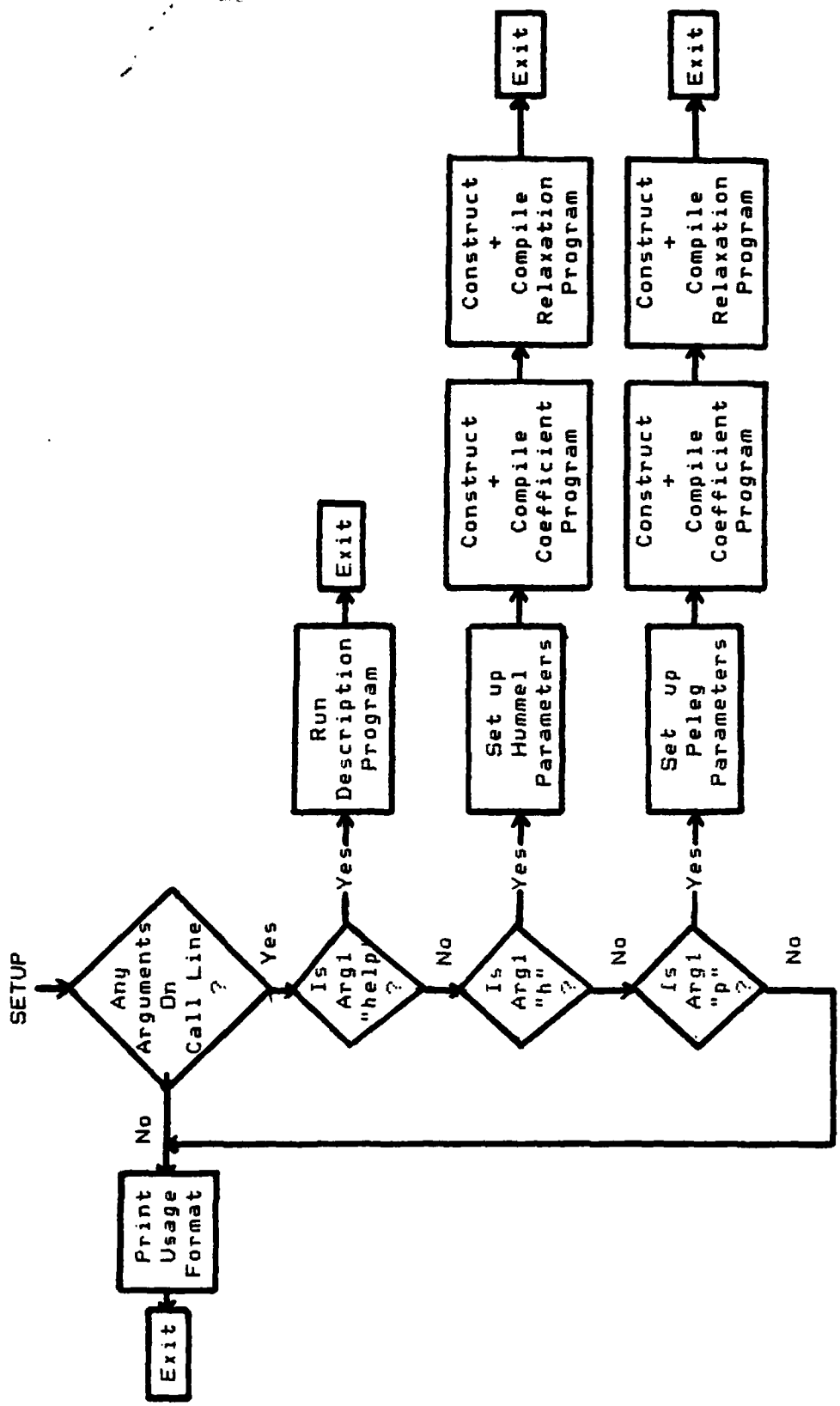


Fig. 2 Package Control Flow

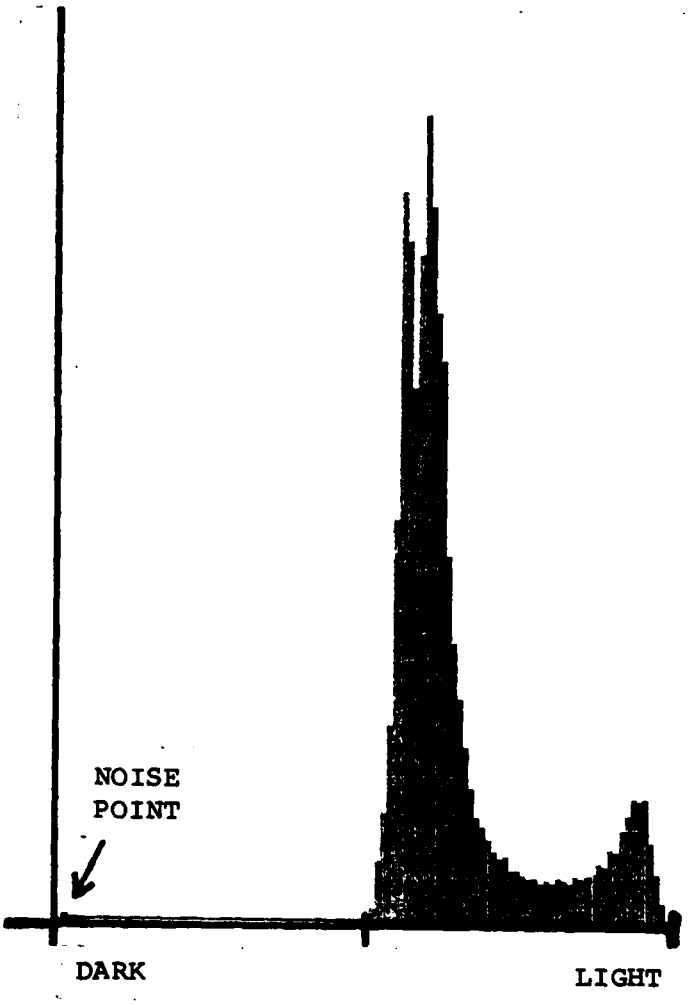


Fig. 3 Initial Classification Error

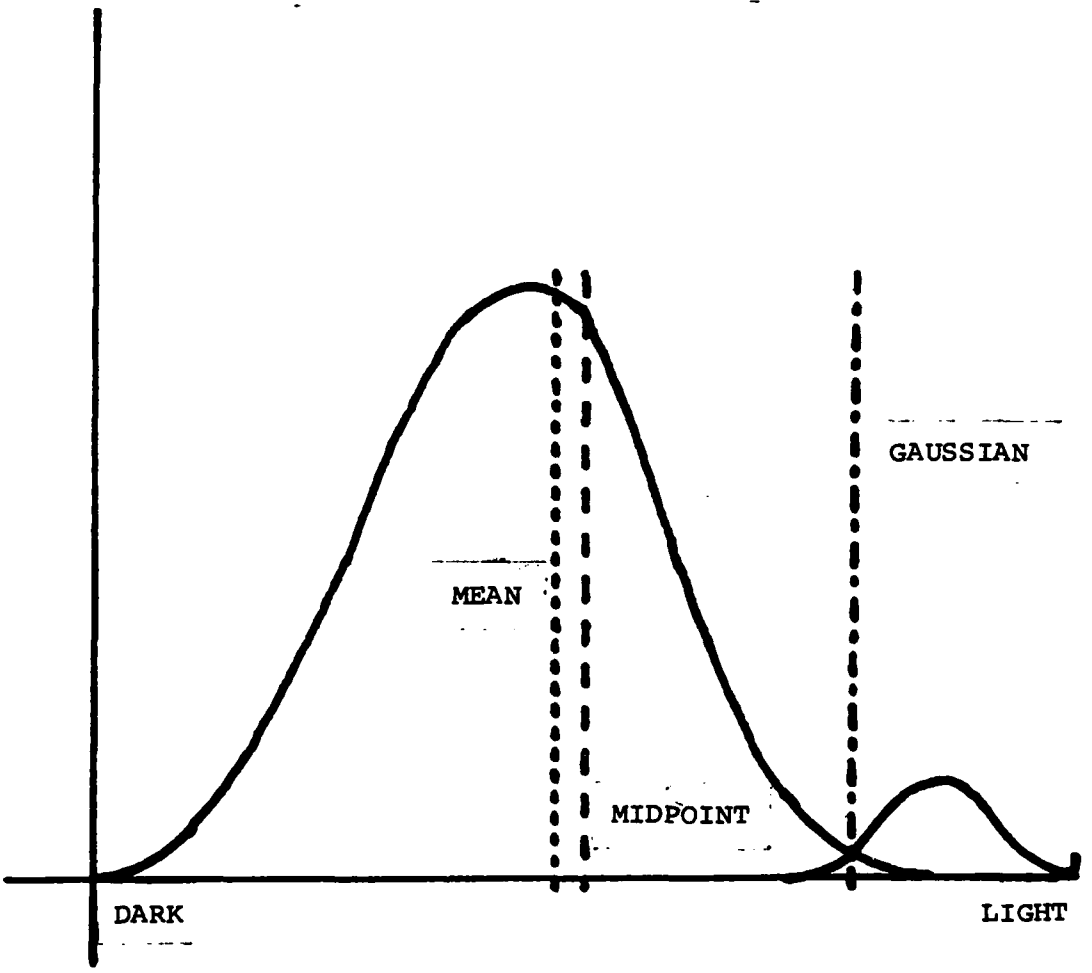


Fig. 4 Midpoint, Mean, and Gaussian Fitting Comparison

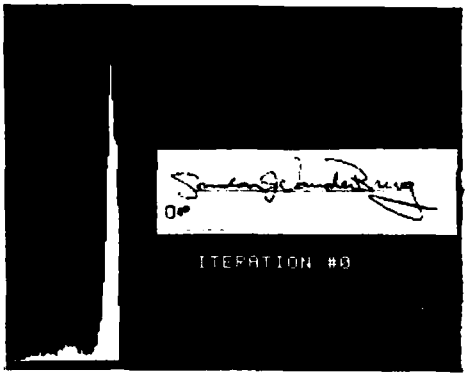


Fig. 5 Original Image -- Signature

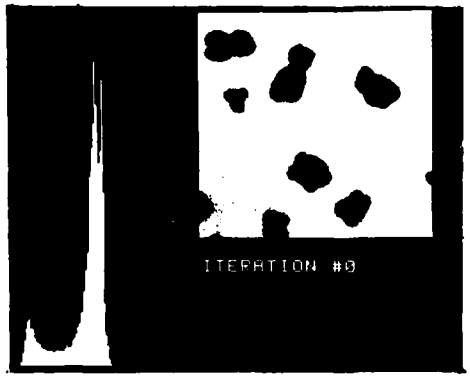


Fig. 6 Original Image -- Chromosomes

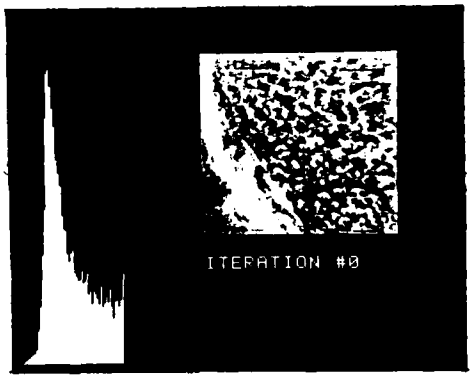


Fig. 7 Original Image -- Clouds

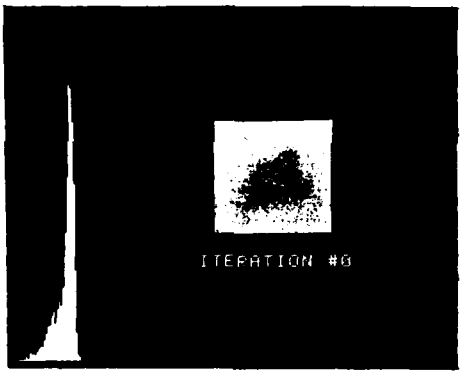


Fig. 8 Original Image -- Tank

Fig. 9 Hummel Method -- Signature

Fig. 10 Peleg Method -- Signature

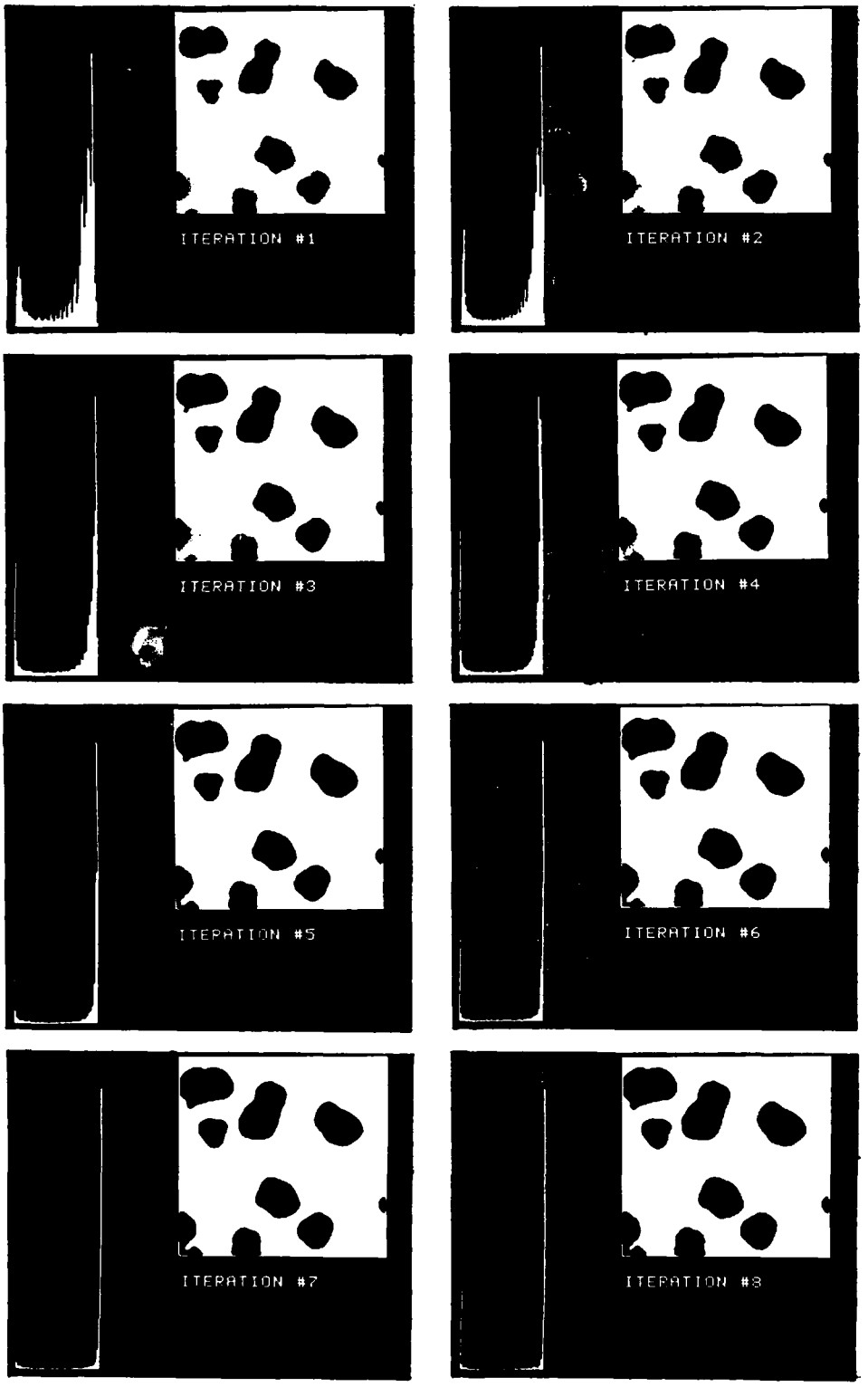


Fig. 11 Hummel Method -- Chromosomes

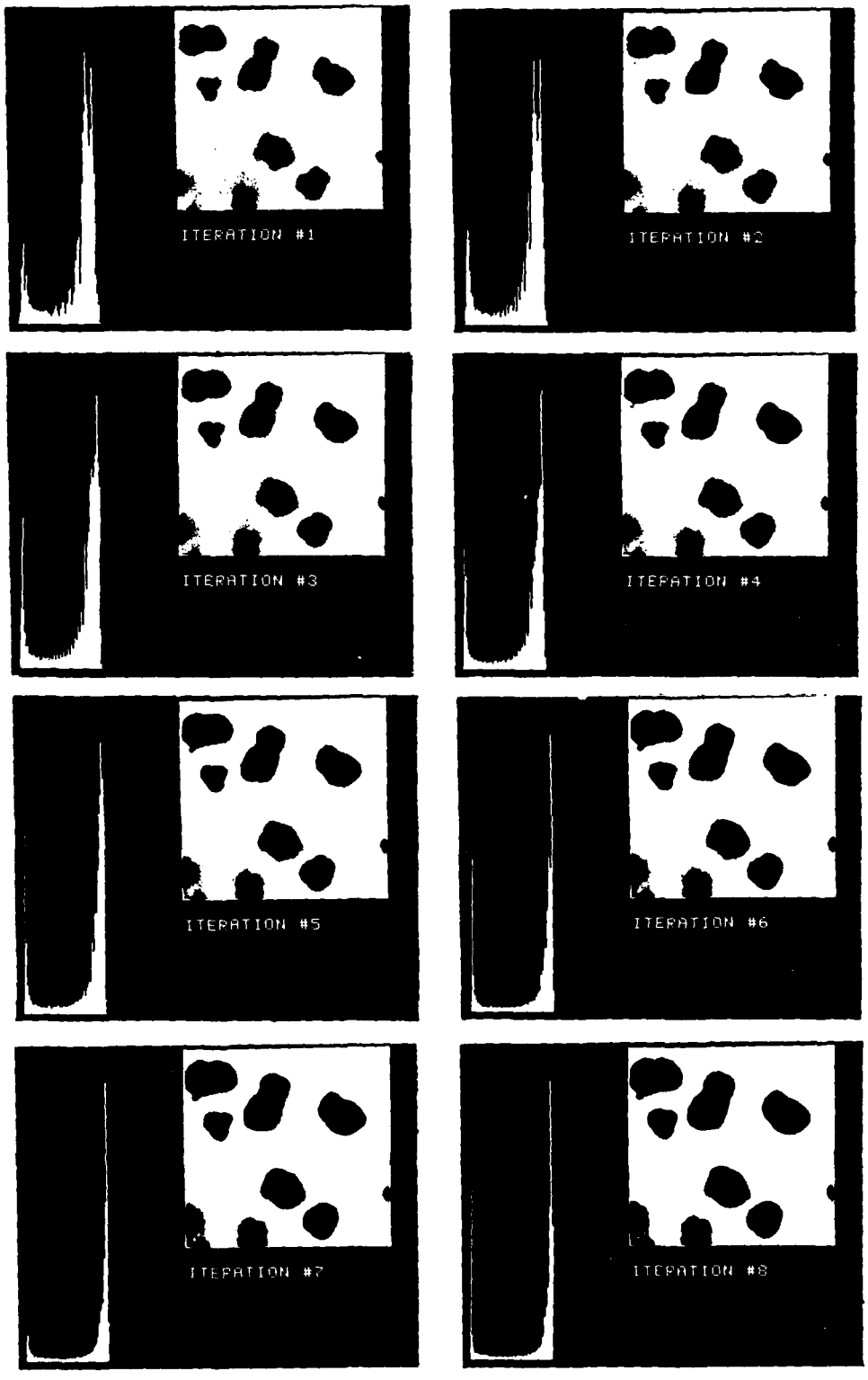


Fig. 12 Peleg Method -- Chromosomes

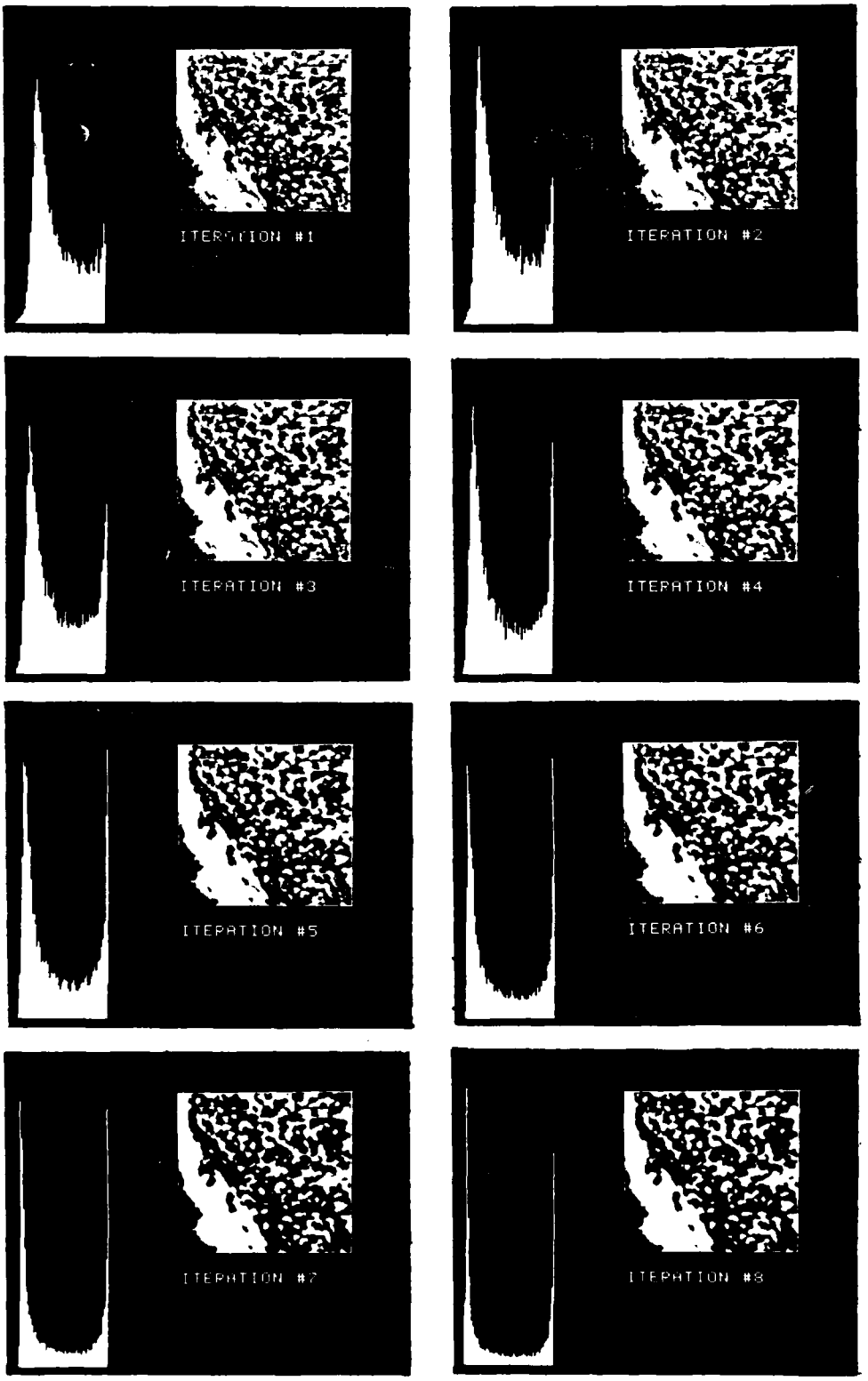


Fig. 13 Hummel Method -- Clouds

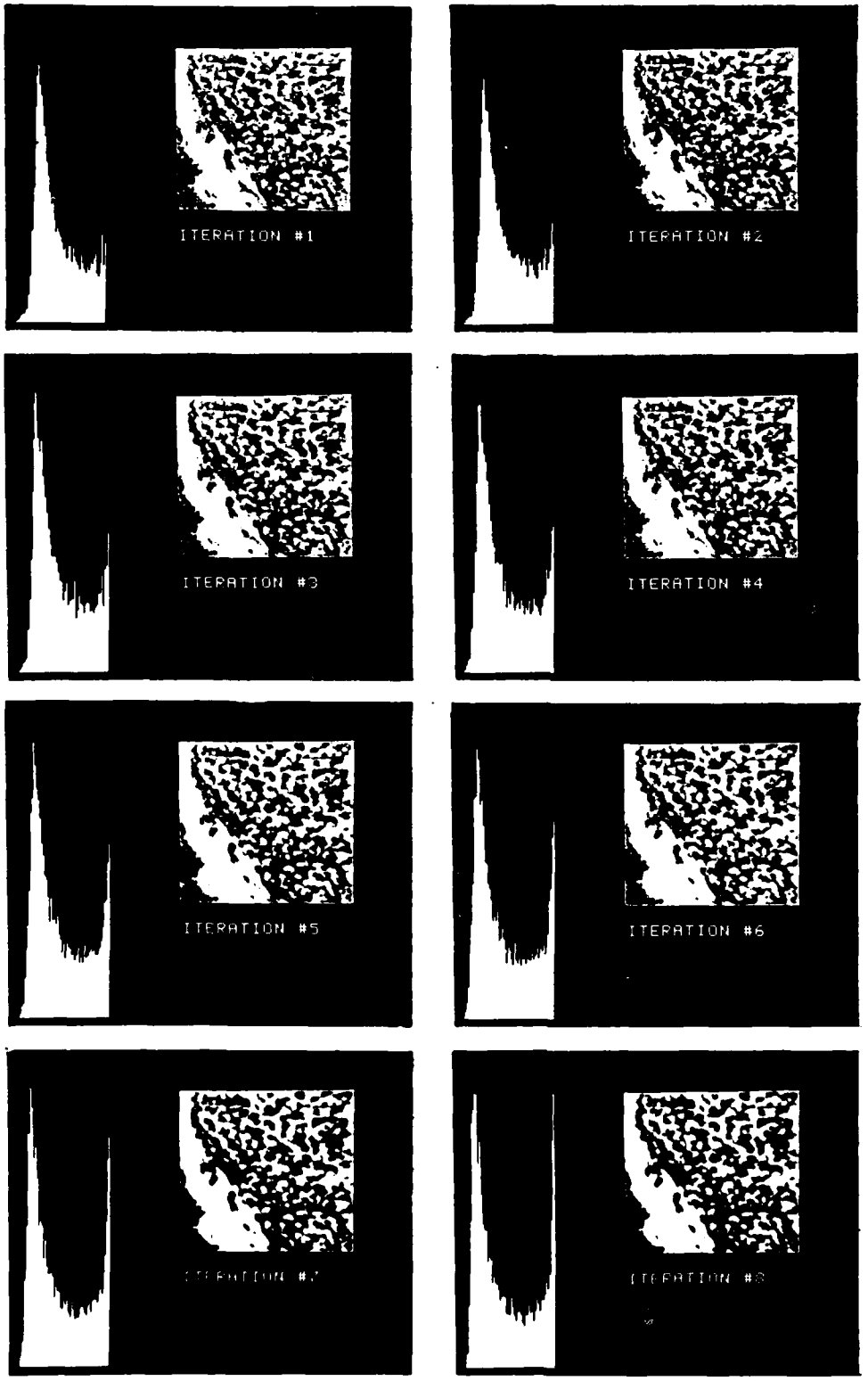


Fig. 14 Peleg Method -- Clouds

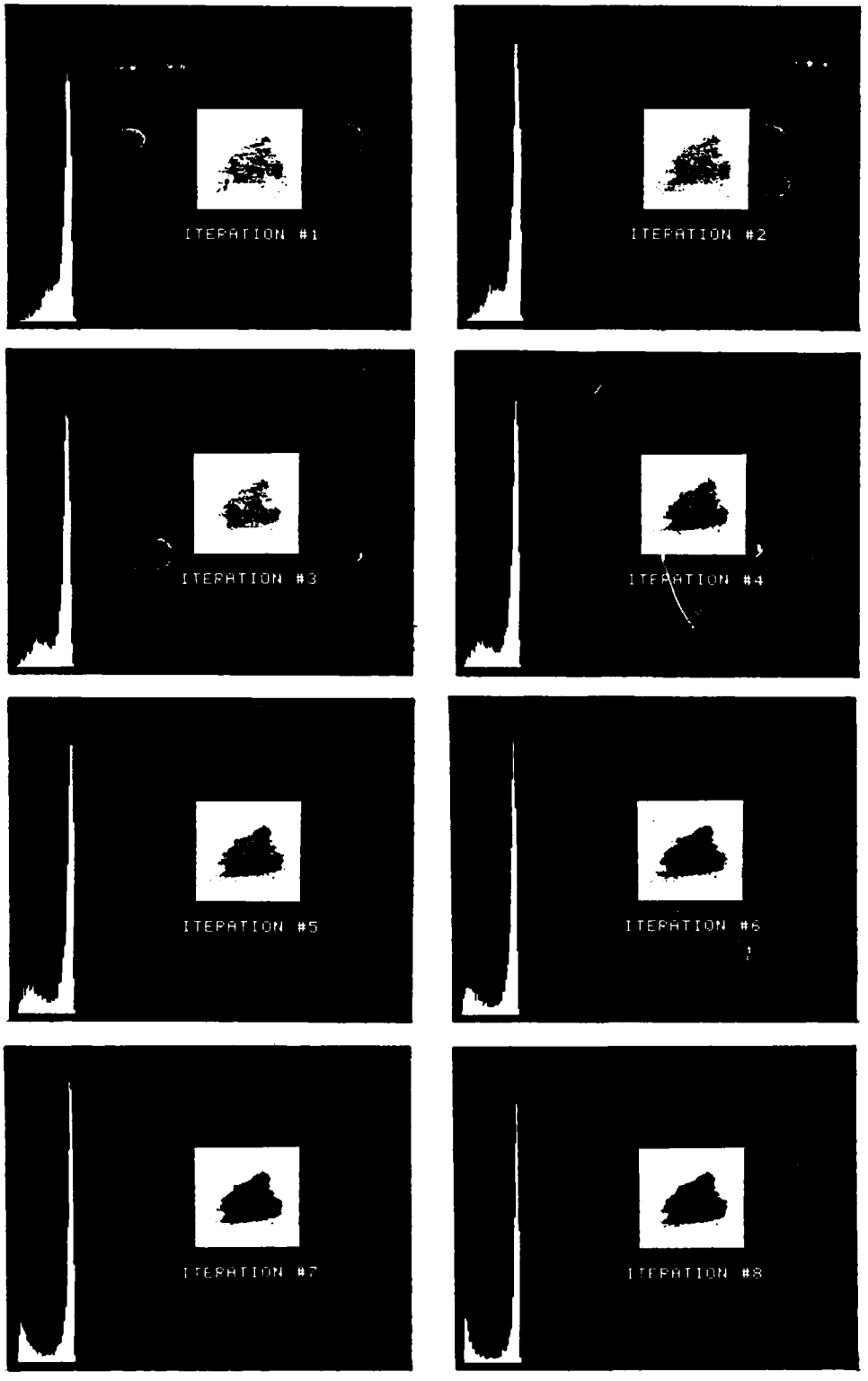


Fig. 15 Hummel Method -- Tank

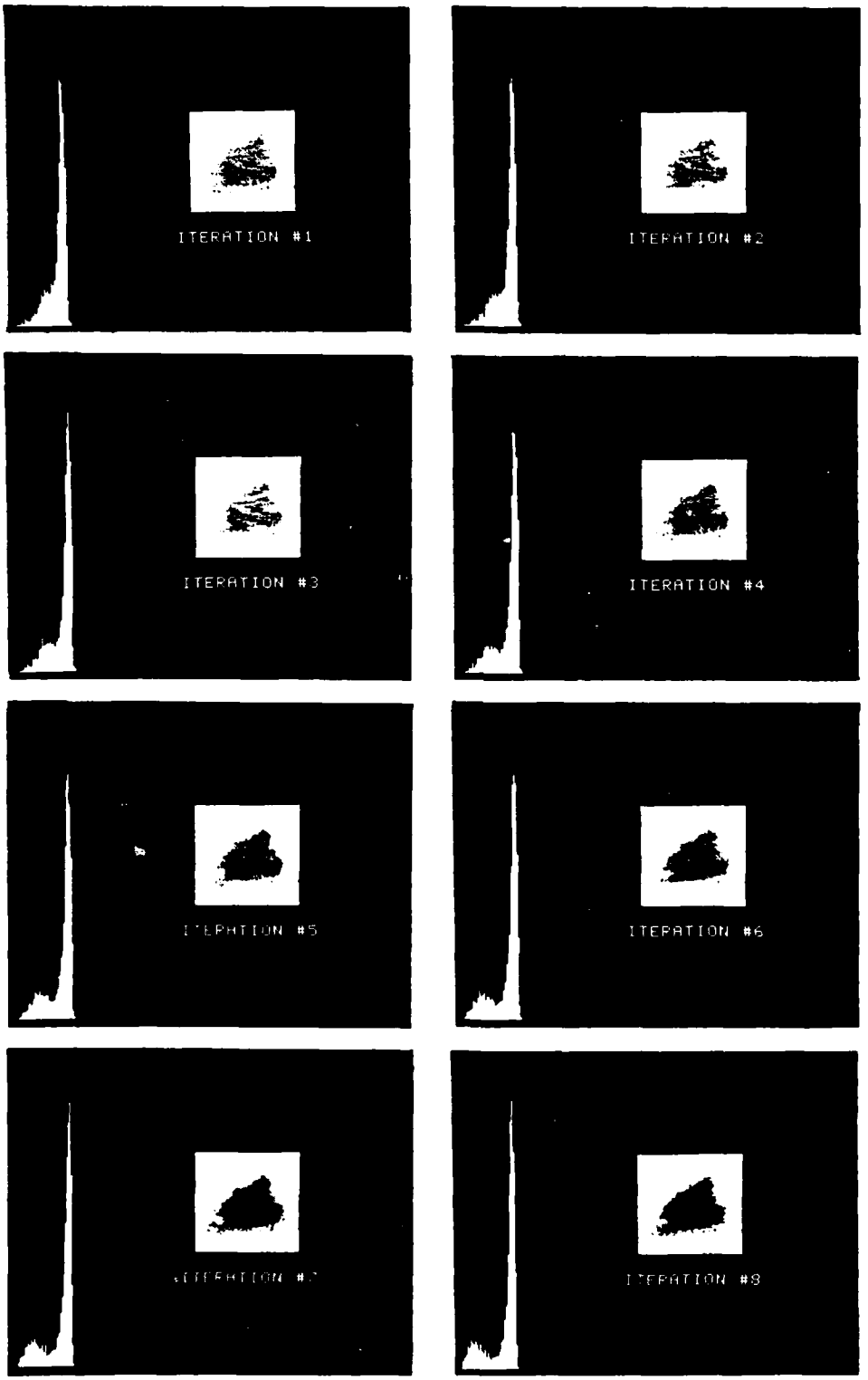


Fig. 16 Peleg Method -- Tank

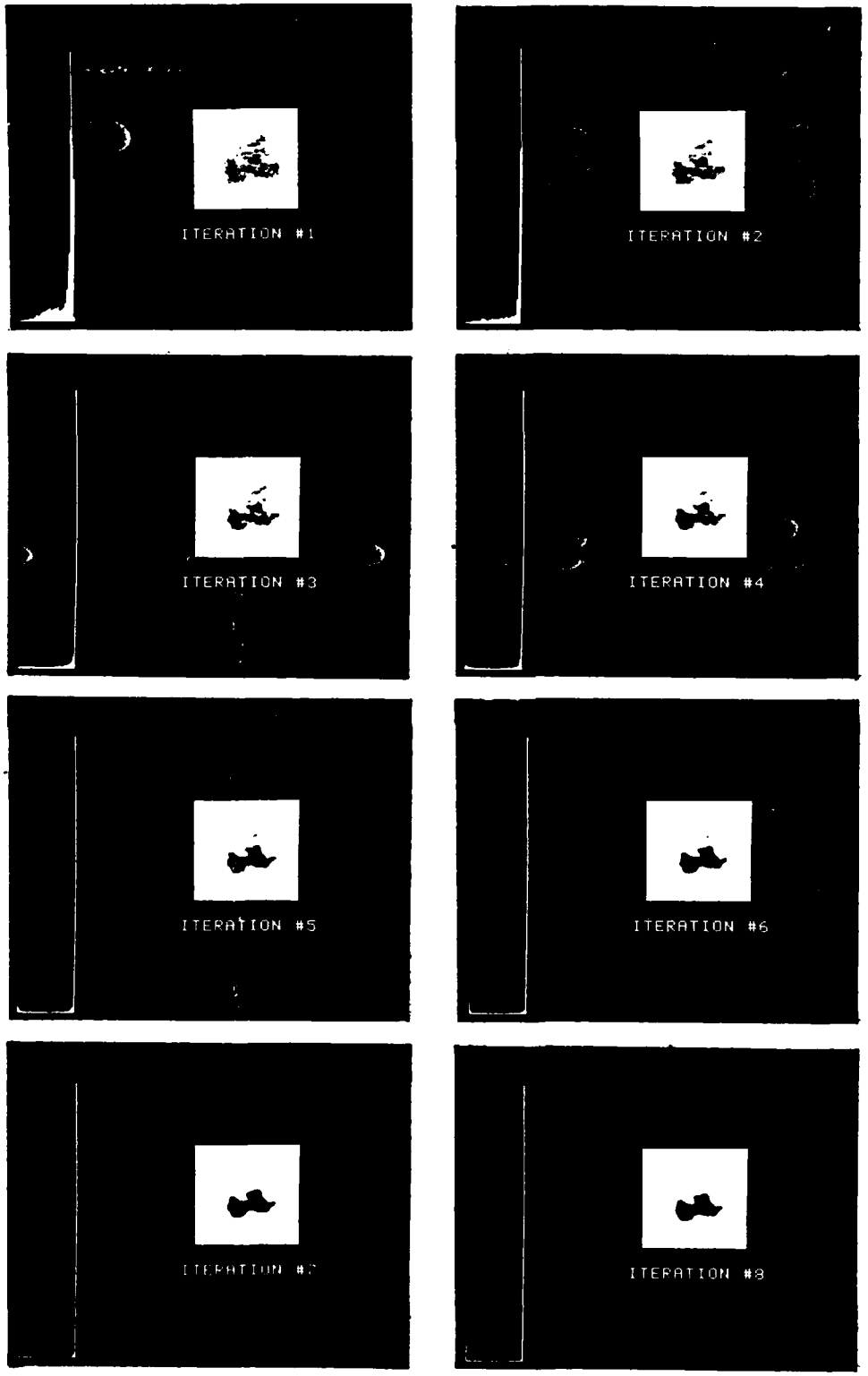


Fig. 17 Effect of Hand Computed Coefficients

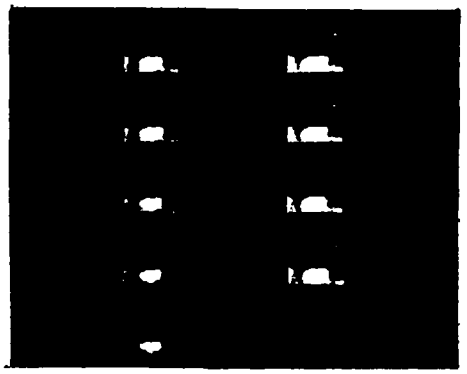
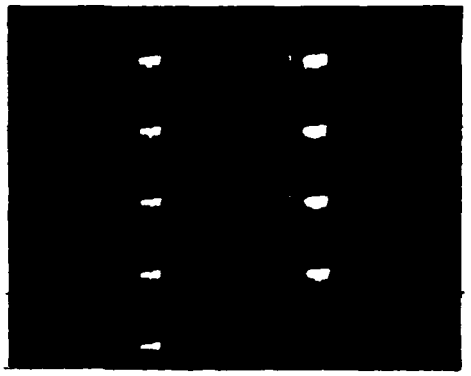
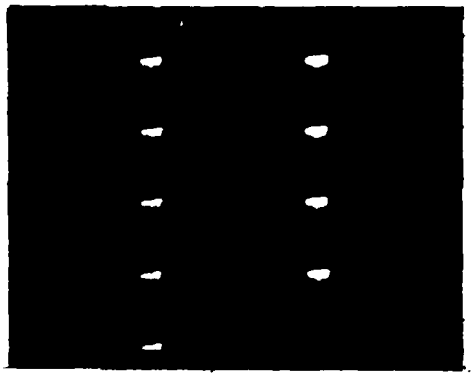


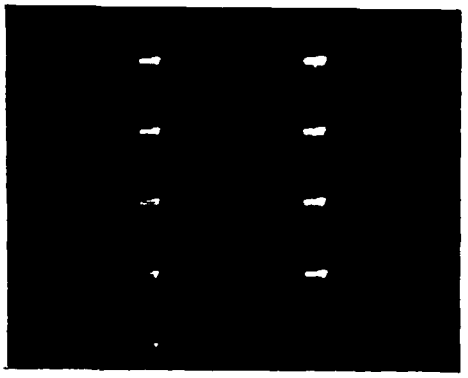
Fig. 18 Peleg Method -- Blob



A = .5



A = .25



A = .01

Fig. 19 Effect of Borderness Initialization

Table 1. Hummel Compatibility Coefficients -- Signature

NEIGHBOR	LIGHT/LIGHT	LIGHT/DARK	DARK/LIGHT	DARK/DARK
A	0.002876	-0.017664	-0.018342	0.082615
B	0.003899	-0.024412	-0.025079	0.104690
C	0.003771	-0.023552	-0.024009	0.101389
D	0.006224	-0.040789	-0.040829	0.146255
P	0.007991	-0.054340	-0.054340	0.174140
E	0.006261	-0.041064	-0.040815	0.146222
F	0.005125	-0.032854	-0.022830	0.097667
G	0.005300	-0.034096	-0.023866	0.100941
H	0.004335	-0.027365	-0.017135	0.078302

A B C
D P E
F G H

Neighborhood of point P

Table 2. Peleg Compatibility Coefficients -- Signature

NEIGHBOR	LIGHT/LIGHT	LIGHT/DARK	DARK/LIGHT	DARK/DARK
A	1.014484	0.915466	0.912368	1.511456
B	1.019687	0.885097	0.882146	1.687839
C	1.019034	0.888908	0.886880	1.660213
D	1.031611	0.815509	0.815343	2.077731
E	1.031803	0.814387	0.815402	2.077386
F	1.025956	0.848513	0.892125	1.629598
G	1.026856	0.843258	0.887516	1.656499
H	1.021911	0.872122	0.917892	1.479215

A B C
D P E
F G H

Neighborhood of point P

Table 3. Hummel Compatibility Coefficients -- Chromosomes

NEIGHBOR	LIGHT/LIGHT	LIGHT/DARK	DARK/LIGHT	DARK/DARK
A	0.027356	-0.028422	-0.027598	0.022038
B	0.032853	-0.035191	-0.034976	0.027082
C	0.028354	-0.029620	-0.029350	0.023264
D	0.033402	-0.035891	-0.035180	0.027217
P	0.043297	-0.049294	-0.049294	0.036034
E	0.032774	-0.035091	-0.035084	0.027153
F	0.028703	-0.030042	-0.029356	0.023269
G	0.032549	-0.034806	-0.034841	0.026992
H	0.026386	-0.027269	-0.027321	0.021842

A B C
D P E
F G H

Neighborhood of point P

Table 4. Peleg Compatibility Coefficients -- Chromosomes

NEIGHBOR	LIGHT/LIGHT	LIGHT/DARK	DARK/LIGHT	DARK/DARK
A	1.105504	0.752869	0.738070	1.613540
B	1.108404	0.746075	0.737014	1.616014
C	1.109018	0.744637	0.741237	1.606122
D	1.109319	0.743934	0.738219	1.613192
E	1.112677	0.736067	0.741704	1.605028
F	1.111016	0.739958	0.743125	1.601700
G	1.113746	0.733562	0.742408	1.603379
H	1.114207	0.732483	0.746930	1.592787

A B C
D P E
F G H

Neighborhood of point P

Table 5. Hummel Compatibility Coefficients -- Clouds

NEIGHBOR	LIGHT/LIGHT	LIGHT/DARK	DARK/LIGHT	DARK/DARK
A	0.020060	-0.056773	-0.060743	0.095686
B	0.020584	-0.058586	-0.061030	0.095992
C	0.020695	-0.058972	-0.059887	0.094765
D	0.020749	-0.059161	-0.060703	0.095643
P	0.021284	-0.061039	-0.061039	0.096002
E	0.021354	-0.061287	-0.059761	0.094628
F	0.021055	-0.060232	-0.059378	0.094213
G	0.021546	-0.061969	-0.059571	0.094423
H	0.021629	-0.062263	-0.058357	0.093097

A B C
D P E
F G H

Neighborhood of point P

Table 6. Peleg Compatibility Coefficients -- Clouds

NEIGHBOR	LIGHT/LIGHT	LIGHT/DARK	DARK/LIGHT	DARK/DARK
A	1.146578	0.867526	0.871107	1.116490
B	1.178525	0.838654	0.839556	1.145005
C	1.152312	0.862344	0.863508	1.123358
D	1.181764	0.835727	0.838702	1.145777
E	1.178059	0.839076	0.839103	1.145414
F	1.154322	0.860529	0.863482	1.123381
G	1.176736	0.840271	0.840126	1.144489
H	1.141028	0 872542	0.872315	1.115398

A B C
D P E
F G H

Neighborhood of point P

Table 7. Hummel Compatibility Coefficients -- Tank

NEIGHBOR	LIGHT/LIGHT	LIGHT/DARK	DARK/LIGHT	DARK/DARK
A	0.005517	-0.018092	-0.018083	0.047380
B	0.005493	-0.018007	-0.018256	0.047764
C	0.005231	-0.017098	-0.018072	0.047356
D	0.006069	-0.020024	-0.019782	0.051116
P	0.007485	-0.025092	-0.025092	0.062172
E	0.005838	-0.019211	-0.019919	0.051412
F	0.005433	-0.017798	-0.017867	0.046897
G	0.005487	-0.017988	-0.018280	0.047817
H	0.005278	-0.017262	-0.018246	0.047742

A B C
D P E
F G H

Neighborhood of point P

Table 8. Peleg Compatibility Coefficients -- Tank

NEIGHBOR	LIGHT/LIGHT	LIGHT/DARK	DARK/LIGHT	DARK/DARK
A	1.027970	0.913512	0.913551	1.267312
B	1.027846	0.913897	0.912762	1.269752
C	1.026499	0.918062	0.913600	1.267160
D	1.030811	0.904727	0.905822	1.291211
E	1.029619	0.908413	0.905204	1.293123
F	1.027537	0.914853	0.914540	1.264256
G	1.027817	0.913985	0.912655	1.270085
H	1.026742	0.917309	0.912809	1.269607

A B C
D P E
F G H

Neighborhood of point P

Appendix

SETUP

SYNOPSIS

setup [[help][h][p]] #labels [#cols #rows]

DESCRIPTION

Setup is a Shell program used to create the relaxation and compatibility coefficient programs according to the needs of the user. The arguments determine the method of relaxation to use, the number of labels in the relaxation label set, and, if something other than the default 3 by 3 neighborhood is desired, the number of columns and number of rows which can contain the desired neighborhood. Setup will construct and compile the programs, leaving them in the user's current directory under the names "*compat" and "*relax", where '*' is either 'h' or 'p' depending on the relaxation method chosen. If the first argument is "help" a short description of the package will be printed. A usage example is:

setup p 2

This will set up both a coefficient computation program and relaxation program for two labels and a 3 by 3 neighborhood following the Peleg formula for probabilistic relaxation.

DIAGNOSTICS

Only a "USAGE..." line if the user mistypes the calling syntax.

FILES

/b/gpstar/par, /b/gpstar/[hp]compat.c, /b/gpstar/[hp]relax.c

COMPAT

SYNOPSIS

[hp]compat probfile comfile [nbrfile]

DESCRIPTION

Compat is a program created by the setup routine which will compute the mutual information compatibility coefficients for either the Hummel-Zucker or the Peleg relaxation formula. The initial letter in the name reflects the method chosen by the user. The arguments to the command specify the probabilistic image from which the coefficients are to be computed, the file to which the coefficients are to be stored, and optionally, a file which contains a nonstandard neighborhood definition. A usage example is:

pcompat tank.p cfile.tank

This will take the probabilistic tank image, compute coefficients according to the Peleg formulation, and place them in the file 'cfile.tank' for later use by a Peleg relaxation program.

DIAGNOSTICS

Only a "USAGE..." line if the user mistypes the calling syntax.

FILES

./hcompat or ./pcompat

RELAX

SYNOPSIS

[hp]relax probfile comfile [[n nbrfile][s N N1 N2...]]

DESCRIPTION

Relax is a program created by the setup routine which will compute one iteration of probabilistic relaxation according to either the Hummel-Zucker or the Peleg relaxation formula. The initial letter in the name reflects the method chosen by the user. The arguments to the command specify the probabilistic image file which is to be replaced by one iteration of relaxation, the coefficient file which is to be used in the computation, and optionally, a neighborhood definition file. If the Peleg method is chosen then the user may also use the 's' argument to specify that the labels should be grouped into N sets of size N1, N2, etc. A usage example is:

prelax tank.p cfile.tank

This will compute one iteration of Peleg relaxation on the probabilistic tank image using the coefficients in the file 'cfile.tank'.

DIAGNOSTICS

Only a "USAGE..." line if the user mistypes the calling syntax.

FILES

./hrelax or ./prelax

DISPLAY

SYNOPSIS

display probfile imagefile #labels

DESCRIPTION

Display will take a probabilistic image file and convert it into a gray level image file for subsequent display. If the "#labels" argument is 2 then the maximum label at each point will be displayed as a range of gray levels depending on label strength. If "#labels" is greater than two the maximum label at each point will be displayed as a constant distinct gray level regardless of label strength. A usage example is:

display tank.p tank.g 2

This will take the probabilistic tank image and convert to a varying gray level image. The gray level image could, for example, be displayed on the GRINNELL display system.

DIAGNOSTICS

Only a "USAGE..." line if the user mistypes the calling syntax.

FILES

/usr/bin/display

DEFNBR

SYNOPSIS

defnbr nbrfile ncols nrows

DESCRIPTION

Defnbr is an interactive program used to define a nonstandard neighborhood for each point. The "nbrfile" argument specifies the file to which the neighborhood definition is to be written. The number of columns and rows which can contain the neighborhood must also be specified. The use of the program is straight forward. An example of calling syntax would be:

defnbr nfile.1 5 3

The user has decided to try a 5 column by 3 row neighborhood. The program will ask which is to be considered the point in the center and whether a certain neighbor is to be considered in the neighborhood.

DIAGNOSTICS

Only a "USAGE..." line if the user mistypes the calling syntax.

FILES

/usr/bin/defnbr

DEFCOM

SYNOPSIS

defcom comfile #labels [nbrfile]

DESCRIPTION

Defcom is an interactive program used to install hand computed compatibility coefficients for each neighbor of a point. The arguments specify the file to which the coefficients are to be written, the number of labels in the relaxation process, and optionally, a file which specifies a nonstandard neighborhood. The use of the program is straight forward. An example of usage is:

defcom cfile.tank 2

The user wishes to define coefficients for a two label relaxation process using the standard 3 by 3 neighborhood.

DIAGNOSTICS

Only a "USAGE..." line if the user mistypes the calling syntax.

FILES

/usr/bin/defcom

IMAGE FILE FORMAT

DESCRIPTION

Each probabilistic image used by the software package contains a header specifying number of columns, number of rows, and bytes per pixel. This header is 6 integer words long:

```
header[0] = 0
header[1] = # of cols
header[2] = 0
header[3] = # of rows
header[4] = 0
header[5] = 4
```

The image data is arranged so that the labels, then columns, then rows change, i.e., the fastest changing value would be the label represented. Each label value should be a floating point number (4 bytes) in the range 0 -> 1. The sum of the label values at each point should be 1. Were the user to declare an array to reflect the format of the image data, the declaration would be:

```
float pimage [NROWS] [NCOLS] [NLABELS]
```

If the original probabilistic image file should not be modified by the relaxation routines then it should be copied, using the copy with the relaxation programs.

COEFFICIENT FILE FORMAT

DESCRIPTION

Each coefficient file contains the raw coefficients and nothing else. Each coefficient is a double precision floating point number (8 bytes). The coefficients are arranged in such a way that the particular neighbor being considered changes slowest, i.e., if one were to declare an array to contain the coefficients it would be:

```
double coeff [NNBRS] [NLABELS] [NLABELS]
```

The first label is considered to be the point's label, the second, the neighbors label. Both hcompat and pcompat will create coefficient files with the correct format.

NEIGHBORHOOD FILE FORMAT

DESCRIPTION

Each neighborhood file contains a 4 integer word header specifying the number of columns and rows in the neighborhood and the coordinates of the point which is to be considered as the neighborhood's center. The neighbors are defined as a column offset from the center point and a row number in the neighborhood. For example, the upper right hand corner neighbor in a 3 by 3 neighborhood around a point would be represented in the neighborhood file as a +1 column offset and as on row 0, hence its entry in the file would be (1,0). If the neighborhood format were declared as an array it would be:

```
int nbrhood [NNBRS] [2]
```

where the [2] specifies both a column offset and a row number.

Bibliography

[1] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans. Syst., Man, and Cybern., vol. SMC-6, pp. 420-433, June 1976.

[2] R. Hummel and A. Rosenfeld, "Relaxation Processes for Scene Labeling," TR-562, Computer Science Center, Univ. of Maryland, August 1977

[3] S. Zucker and R. Hummel, "Computing the Shape of Dot Clusters, I: Labeling Edge, Interior, and Noise Points," TR-543, Computer Science Center, Univ. of Maryland, May 1977

[4] A. Lev, S. Zucker, and A. Rosenfeld, "Iterative Enhancement of Noisy Images," TR-445, Computer Science Center, Univ. of Maryland, March 1976

[5] J. Eklundh, H. Yamamoto, and A. Rosenfeld, "Relaxation Methods in Multispectral Pixel Classification," TR-662, Computer Science Center, Univ. of Maryland, July 1978

[6] A. Rosenfeld, "Iterative Methods in Image Analysis," Proc. IEEE Conf. on Pattern Recognition and Image Processing, pp. 14-18, June 1977

[7] S. Peleg and A. Rosenfeld, "Determining Compatibility Coefficients for Curve Enhancement Relaxation Processes," IEEE Trans. Syst., Man, and Cybern., vol. SMC-8, pp. 548-555, July 1978

[8] S. Peleg, "A New Probabilistic Relaxation Scheme," TR-711, Computer Science Center, Univ. of Maryland, November 1978

[9] D. Ritchie and K. Thompson, "The UNIX Time-Sharing System," CACM, vol. 17, Number 7, July 1974

[10] S. Bourne, "The UNIX Shell," Bell System Technical Journal, Vol. 57, No. 6, Part 2, pp. 1971-1990, July 1978

[11] A. Danker and A. Rosenfeld, "Strip Detection Using Relaxation," TR-725, Computer Science Center, Univ. of Maryland, January 1979

[12] G. Fekete, "Relaxation, Evaluation and Applications," Master's Thesis, University of Maryland, 1979

[13] A. Danker and A. Rosenfeld, "Blob Extraction by Relaxation," TR-795, Computer Science Center, Univ. of Maryland, July 1979

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A086 100	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A GENERAL-PURPOSE SOFTWARE PACKAGE FOR ARRAY RELAXATION		5. TYPE OF REPORT & PERIOD COVERED Technical rept.
		6. PERFORMING ORG. REPORT NUMBER TR-839
7. AUTHOR(s) Russell C. Smith		8. CONTRACT OR GRANT NUMBER(s) DAAG-53-76C-0138
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Center University of Maryland College Park, MD 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 12 186
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Night Vision Laboratory Fort Belvoir, VA 22060		12. REPORT DATE December 1979
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 15 DAAG53-76-C-0138, DARPA Order- 3206		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Pattern recognition
Image processing
Relaxation
Thresholding

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Probabilistic relaxation as an image processing tool is becoming increasingly common. This report examines two versions of probabilistic relaxation, that initially proposed by Hummel, Zucker, and Rosenfeld, and the version recently introduced by Peleg. A software package is presented which allows either method to be applied to probabilistic images quickly and easily. The package makes full use of the power available under Bell Laboratories' UNIX operating system

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

411074

Abstract (continued)

running on a PDP11/45 computer. Modular in form, it frees the researcher from the tedium of hand coding relaxation processes for each variation of relaxation tested. The application of relaxation to a threshold-like gray level modification scheme demonstrates the utility of the package.